CHICAGO USERS GROUP PRESENTS

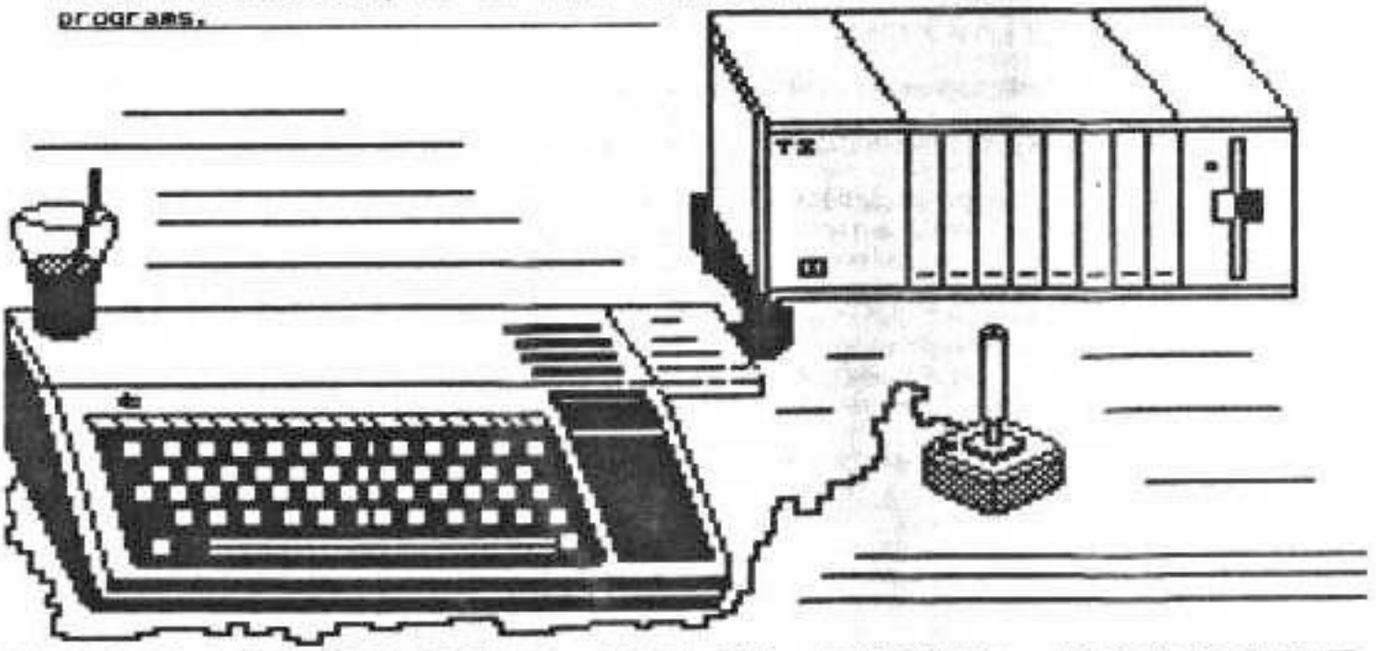
CHICAGO TIMES

NEWSLETTER OF THE CHICAGO TI-99/4A USERS GROUP

THIS IS OUR ANNIVERSARY!!

FEB. 28, 1986
EDITOR: Carole Goldstein

The Chicago TImes is published 10 TImes a year, monthly, except during June and July. Chicago TImes is not affiliated with Texas Instruments in any way and is supported only by its subscribers and advertisers. Subscriptions are free with membership in the Chicago TI99/4A User's Group. The Chicago TImes is also distributed free to any other User's Group that wishes to reciprocate. Articles contained herein may be re-printed by another User's Group Publication provided credit is given to the Chicago TImes as the original source of the article. Comments and letters are welcome, as is the submission of original articles and programs.



WHAT'S WRONG WITH THIS PICTURE

THE MARCH MEETING

will be held on Saturday Mar. 1, 1986 from 1:00 to 3:00 in the Fireside Lounge at Triton College. The meeting will feature the GRAM KRACKER, a color printer hooked up to the TI and a question and answer session.

IN THIS ISSUE...

THE DISASSEMBLY	Dave Wakely	Page 3
THE MAIN EVENT	Sam Pincus	Page 6
FULL DUPLEX	Irwin Goldstein	Page 7
LETTERS TO THE EDITOR		Page 8
MEMBERSHIP CHAIRMAN SPEAKS	Don Jones	Page 9
SOFTWARE REVIEW	Jack Topham	Page 11
ADVANCED ASSEMBLY ROUTINE	Nick Iacovelli	Page 13
FROM THE OTHER ORPHANAGES	Jack Topham	Page 18
ALL 'BOUT C		Page 19
TECH CORNER		Page 24
RAM BUGS?	Texincia Lubbock	Page 27
REMarks	Carole Goldstein	Page 28

BULLETINS:

The PASCAL SIG will meet at 11:00 the day of the meeting.

The BUSINESS SIG will meet after the regular meeting.

A new SIG called SO WHAT DO I DO NOW? will start after the meeting. This is for our members who are just putting their systems together and need to know the absolute beginning basics. There will be people there to explain the parts of the system and to answer your question. If you are new into the world of computers and have your system all set up and have been asking the question "AND WHAT NOW?", then please come to the new SIG.

MEETING DATES FOR THIS COMING YEAR ARE AS FOLLOWS:

MARCH 1	SEPT 6
APRIL 5	OCT 4
MAY 7	NOV 1 (TI FAIRE)
JUNE 7	DEC 6

THE DISASSEMBLY: Dave Wakely

A new book, a nightmare, a fix, exploding myths, and other Things:

It was while looking through some old papers that I realized I had missed the anniversary. Back in the first week of January it was exactly five years since I had purchased my first TI. In my hand was the original receipt from Computerland in Niles for $300 for their demo TI-99/4, plus $100 for a side-car disk controller I would not be able to use for another 10 months. Somewhere in the same envelope was a receipt from U.S. Micro Sales in New Lennox for $359 for a disk drive, case, and power supply, dated November, 1981. I seem to remember talking Computerland out of their last remaining Demonstration, Diagnostics, and Video Games I modules, but what's this? It's a receipt for $69.95 for the Football module! Expensive? Not at the time. A TI owner was glad to get his/her hands on literally anything back then.

What a five years it has been since then! Looking back at the original announcement of that "all-TI" magazine which this column refuses to ever again mention by name; through the release of the 99/4A; the great Extended BASIC shortage; the general inaccessibility of the inner workings of the machine; it now occurs to me that there were some very clever Psychologists at work at Texas Instruments. I have sometimes suspected that precisely because it was so difficult to get the machine to do exactly what we wanted it to do that so many took this as a challenge to do it better, faster, and cheaper.

All of the above, however, is the past. Those interested in it might want to look at the last chapter (actually the last appendix) of the new book "The Orphan Chronicles", edited by TI Forum Sysop Ron Albright and published by Miller's Graphics (hint: Hunter Electronics has it for $9.00). This chapter came about when Albright wrote to me last summer asking for any details at all about our group. At a time when I probably should have been working on this column, I managed to get off about six pages of rambling reminisences detailing the history of the Chicago TI-99/4A User's Group, and then promptly forgot about it. Two months later Ron called and informed me he was including most of the letter in his forthcoming book about the TI "phenomemenon", both past and present. I believe he said the chapter was to be titled "The Legacy of the Super Group", but I am unsure if he actually used it becuase I don't have a copy of it. No, I asked for nothing for my contribution when I found out my letter would be included in the book, and yes, it would have been nice if either Albright or Miller had sent me a copy. I now know how some of those "Freeware" authors feel. I also apologize to all of you who have worked hard for the group over the past years whom I didn't mention in the chapter.

Oh, NO!: I suppose it was inevitable that this would happen. On several occasions over the past few months I have mentioned my pleasure with the "49-99" keyboard from Degris Electronics. That's the one with the extra FCTN key to the left of the "A" key. Imagine my dismay when, right in the middle of writing last month's column, the console started locking up on me. This is called "having your worst nightmare realized." None of the usual fixes (checking all cable connectors and the module port) could induce the console to desist from locking up about every 20 minutes or so. "So what?", you are probably saying to yourself. "Take the offending console down to the TI Exchange Center in Arlington Heights, plunk down the $30 or so, and walk out with a new console." Not so fast

there. I do not want Texas Instruments to take possession of my modified keyboard, or to refuse an exchange because of "tampering".

One solution might be to open up the case, remove the "49-99", put back the original keyboard, and then do the swap. The ensuing conversation at the Exchange Center might then go something like this:

Me: Uh, I would like to exchange this non-functional 99/4A for a working one.

TI: Fine, sir. That will be $30. May I see the console? Hmmm...What exactly is this hole in the keyboard cover next to the "A" key?

Me: Uh, that. Well, you see, it's...uh where a small meteorite crashed through my home and into the keyboard, neatly taking out a small piece of plastic. I don't think that has anything to do with the console not working, though.

TI: Excuse me a moment sir, while I go talk to my supervisor.

Now, there are reasonably understanding people down at TI, right? What if I just take it to them as is, and tell them that I want this particular keyboard back after they fix it?

Me: Uh, I would like to turn in this non-functional 99/4A for repairs, and then get it back again.

TI: Yes sir, we can do that. May I see the console? Hmmm...hey, what the $@' is this? Where did you get this console with this extra key? You didn't tamper with it, DID YOU?

Me: Uh, that. Well, you see, its...uh a slight "user friendly" modification that was done to the keyboard to make typing easier. I know TI didn't sell them this way, and that's not what's wrong with the console, but that shouldn't effect your repairing it, right?

TI: Excuse me a moment sir, while I go talk to my supervisor.

Of course, I have also heard it said that TI doesn't really care what condition the returned unit is in, and this has given me a few ideas on how to handle the problems outlined above. For example, before I got into computers I used to have other hobbies. I seem to recall a particularly heavy bowling ball which must be somewhere in the basement. Suffice it to say that an "accident" occuring from a height of, say, three feet vertical to the left side of the keyboard should remove any evidence of a small, missing piece of plastic. To paraphase what they used to say during Viet Nam, "We had to destroy the console to exchange it."

What I will probably end up doing is calling the Exchange Center anonymously and asking some hypothetical questions. ("What if I just happened to have a modified console which went bad...") If I don't like the answers I get I will probably call the TI parts department and purchase a new top, use it to replace the old one as well as changing back the keyboard, and then do the exchange, reversing the process with the new one. A lot of work, but the 49-99 keyboard is worth it. As Murphy's law would have it, I finally got used to typing on it when it went down, such that the little finger on my left hand still occasionally

goes "thwack!" on the plastic next to the "A" key on my backup console. Sigh (and ouch).

RECOVERING NICELY, THANK YOU: If there was anything good about the whole experience, it is that I learned a few things about TI Writer. As I mentioned, the lockups were occuring while I was writing last month's column. Despite the best of intentions to back up frequently, at one point several pages just vanished (the road to lost text is paved with such intentions). Veteran TI Writer users know that the "RE" (Recover Edit) function will in fact NOT recover your text once you have left the TI Writer environment. Like if your console has locked up and you must turn it off and then back on again. Well, operating on a hunch gleaned from experience with Extended BASIC, I did not turn off the expansion box, and a little experimentation with some Editor/Assembler BASIC peeks returned most of the text to me. I now have a program tentatively titled "REAL_RE" (REAL Recover Edit), which can recover text from the PE box memory even from a console restart. In its present state I am probably the only person who can use this program, but I am cleaning it up and will submit it to Carole for the newsletter, as well as to the library, when ready. I can tell you that some VERY strange things are in memory expansion when TI Writer is using it.

ANOTHER MYTH EXPLODES: Somewhere or other recently, I came across an old rap on the 99/4A keyboard. You have all heard it, "The TI keyboard has all the keys squashed together," or something similar, implying that this isn't a "real" keyboard. While I am by now quite comfortable with the keyboard (except for my missing FCTN key, of course) I half believed this baloney myself. That is, until I took a long look recently at the keyboard on an IBM PC. Users of this machine know that the color of the alpha keys is slightly different from the color of the function and numeric keys on the IBM. It occured to me that it was approximately the size of the keyboard on my TI. Some quick work with a ruler confirmed my suspicions. In fact, the alpha portion of an IBM keyboard is EXACTLY the same size as the alpha keys on the 99/4A! I used the ruler to measure across the number keys from "1" to "0", and across a row of alpha keys ("Q" to "P") on each computer, and each time the measurements were the same. Vertically, from the bottom of the space bar to the top of the "6" key, and even diagonally, from the left lower corner of the "Z" key to the top right of the "P" key also yielded the same distances. The only discrepancy I could find is that the space bar on the TI is about one half inch shorter than the one on an IBM. Big deal. What this means is that typing letters on the TI requires the same "reach" as it does on an IBM PC. So no more talk about the keys being "squashed together", ok? And unlike what they tell you on TV, -kids, it's ok to try this trick at home!

SUBROUTINES: My congratulations to the L.A. User's Group. Not on their upcoming T.I. Fair, but on getting an announcement of it into the March Computer Shopper. In 1983 we tried to get an announcement into the Shopper, without success. In 1984 we wrote both to the magazine as well as directly to TI columnist Randy Holcomb, practically begging for any type of mention about our Faire. For our efforts we got zilch. Last year we didn't even try...Speaking of Faire(s), it seems that everyone is holding one now. I have seen announcements from New Jersey and elsewhere, but my favorite is for the upcoming New England "Fayuh"...Now which TI User Group was it that did the very first "TI Faire"??...Thank you Jan Jankowski for the presentation at the February meeting, I assume the people asking the intelligent questions were all Ham operators

themselves...Those Degris brothers are at it again. It seems they just can't leave the TI console alone. Details next month...I have decided not to try to become the Mike Royko of TI columnists after all, instead I am aiming to become the Jerry Pournelle of TI columnists. I figure to be there Real Soon Now...

THE MAIN EVENT: Sam Pincus

I was gratified at our January meeting to see the number of members who were willing to volunteer for future events. With guys (and gals) like you, my and Dave's job is going to be easy. Jan Janowski's presentation was pretty interesting. Its too bad that the communication band went "dead" on him but those of us who live with Murphy's Law (what can go wrong ...WILL GO WRONG) know that these things happen. We hope that we can continue to provide you with these kinds of interesting and wide-ranging presentations. I've updated our calendar of events with our latest ideas. Don't forget- if you want to volunteer for a presentation (or have an idea for one), let me or Dave know at our next meeting.

March: This month is being set aside for question and answers. Our group's technical guru's will answer all (I hope) of those questions about the TI that have been bugging you. Another portion of the meeting will be hosted by our own Hank Ellerman who will explain what some group members are doing with the "Gram Cracker" package. Burke Crane will demo his color printer - Mike Schube has promised to bring along his video camera so that those of you in the back row will be able to see the printer in action. April: This is the last month to turn in your entries for the annual programming contest. Remember, this year we will have a Junior's division cont est (for programmers 12 and under) as well as our regular contest. The presentation will be on Bulletin Boards. Todd Kaplan will talk about the features of Compu-serve, Tom Kirkpatrick will discuss the Source, and John Behnke and Butch Goldstein will show the latest features of our own board. May: We will be demonstrating all of the winning entries in our programming contest.

June: This is the month of our big hardware/software raffle. Here's your chance to pick up a used piece of equipment at a reasonable price. It's also a good time to help the group by donating some of the stuff that you don't use any more to us so that it can be raffled off. Please give your donations to me or Dave before the June meeting (hopefully in May) so that we can establish a minimum bid price for your donation. We recieved 2 donated packages already with a third one promised. Hope there are more of you out there who will donate material! Summer: We are hoping to set up a summer picnic at Schiller Park for July 27th. We will keep you posted on progress as the year goes by. (P.S.- We will probably be asking for volunteers to help formalize the activities once spring finally breaks out.)

September: Stu Olson will be demonstrating how he builds his "SUPERCART" containing 8K of memory from old TI cartridges. Also, the DeGris brothers will show how they set up their special Command module by rewiring old TI cartridges. Should be an interesting meeting. October: Bob Nater has promised us a presentation on the "C" programming language. We are now on the look-out for a volunteer to talk about PASCAL/TI FORTRAN/ or FORTH.

November: The 4th annual TI FAIRE!!!

December: Election of officers and Swap-a-rama. I'll continue to update this list as time goes by. Don't forget, if you have any thoughts on what you would like to see presented or want to make a presentation yourself, just let us know at any meeting. We would especially like to talk to any of you who are doing exotic things with your TIs' ers. Our group's technical guru's will answer all (I hope) of those questions about the TI 12 and under) as well as our regular contest.

FULL DUPLEX: Irwin Goldstein

This month the bulletin board has received a nice addition to its already impressive array of hardware. The new member of the BBS equipmen family is a 512K memory card from Myarc being utilized as a ram disk.

The Ram Disk looks exactly like any other expansion box perifial card; but it acts exactly like a standard disk drive-only much faster. Speed is the big advantage to using a Ram Disk system. Files can be saved and retrived almost instantly. TIhis is evidenced when logging onto our BDS, when reading messages, and especially when saving messages. Comments from users have been very favorable towad the continued use of the device.

Originally, we had planned to utilize the Ram Disk strickly for use on the download section anticipating the mass storage of over 100 downloads. However, we have since learned that of the 512K memory available for use, 80K is devoted to a print spooler(which is not needed for the BBS) and 32K is used in place of the memory expansion card-which will now be auctioned at our June meeting.

If my calculator is working right, that leaves 400kilobits (or 1600 sectors) for disk storage. A double sided, double density standard disk drive has 1440 sectors available. Thus, we have a net gain of 160 sectors of storage, not realy enough to make any measurable difference in the number of download programs capable of being stored on disk.

What we do gain however is speed and that is measurable to the extent that valuable board time is not wasted in waiting for the disk drives to log you on and set up messages bases.

To see just what I mean by all this verbage, give the board a call for yourselves and see the difference in access times. Then leave me your comments as to just what you think of it. you may leave your remarks either on the Open message base or you may leave me a private message in the (W) section. I'd like to know YOUR thoughts on the subject.

We will be building up the Open and Private sub-boards soon. We now have room since we currently are operating 2 double sided, double density disk drives and one Ram Disk drive. Watch for the expansion.

To call the BBS, dial 966-2342 anytime of the day or night any day of the week. we never stop working!

LETTERS TO THE EDITOR

There has been a lot of discussion about Fairware, Freeware, Honorware, Previeware, and all the other software that, by any other name, has the characteristics of being available at no charge to the user, so that the user might evaluate and review it at their leisure. There is a proviso to this software that usually makes the user honor bound to send the programmer the amount of payment asked in the documentation to the program.
This seems a very equitable, and inexpensive way to distribute quality software at reasonable prices. The problem with this method of distribution is of course the honor system it is based on. Most people, I feel, are basically honest, and tend to do the right thing if properly motivated.

The motivation in the case of this type of software comes from the ease of procurement, the ability to preview, the prime quality of the average offering, and the VERY reasonable cost to the purchaser. The programmers that offer such fantastic bargains have been less than enthusiastic about continuing with this system, not because of the lack of response to their program's performance, but to the lack of reimbursement for their efforts.
Naturally, if it was financially worthwhile for these programmers to write this type of software, they would redouble their efforts to make more product available to the public. Unfortunately payments are meager, and few.

Now I would like to present a plausible explanation to excuse the majority of ethical software users nationwide. Software of this type is being downloaded from electronic bulletin boards from all over the country. I have seen kids, teenagers mostly, downloading prodigious amounts of software from BBS systems just to make sure that they are not caught short when it is time to trade serious business and applications software for the games and fun type programs that they enjoy. Most of the software used for this purpose remains unused and the disk documents unread. Yes I know, there can't be that many kids across the nation doing this kind of trading to make such an imbalance in the disparity between use and possesion of Freeware, (WANT TO BET?). In addition, these kids (TEENS) usually cannot afford even the pittance freeware programmers are asking, because they have no means for generating income as it is needed. Can't you hear dear old Dad when his progeny comes to him and says: Gee Dad, I just downloaded three programs that I think are just terrific, and I would like to buy them. Don't worry they're Freeware programs and they don't cost very much, they're only twenty bucks apiece, sixty for the three. What would you're reaction be?
Perhaps you knew nothing about computers except that the one your kid has, is a constant drain on your savings account, and makes your phone bill a candidate for comparison to the national debt. I am sure your reaction would be to tell your child to wait a minute while you get your checkbook. Right?

ANON E. MUSS

2/4/86

Dear Editor,

I am writing this letter to voice my opinion of your newsletter.

First, let me say that for the most part it is a good newsletter. You touch base with most interest groups. Personally, I would like to see more informative assembler language articles, but this is not a complaint.

More often than not, I receive my copy of the newsletter on the Monday or Tuesday following the meeting. I cannot attend all of the meetings, but if I had an idea of what was going to be covered ahead of time, I might be able to make special arrangements once in a while.

The main reason I am writing this is to voice my opinion of your column called "Ram Bugs". Last month's column had no place in a family oriented computer club publication. What did that dribble have to do with family computing? This month's column (2 1/2 pages loooong) was a complete waste of space. The "freeware" topic was a topic that was beat to death last month. But to spend almost 2 pages whining about someone woh left her (or him) a message on the board is ridiculous. As editor, do you really think anyone else cares? I know I don't. I work with 6 other

members of your TI club and it seems they all agree.

Your newsletter will improve immensely once you quit printing this garbage and write more about the computer.

Sincerely,

Robert Kolling
Member #033

Dear Robert:

I am sorry you find Texincia such a turn-off. I do believe you are in a minority. Most of the people I talk to turn to her column first thing when they receive their newsletter.

As for the mail, well, what can I say. There are some things I have no control over. I do my best to get this newsletter out on time.

But as for not knowing what is happening at the meetins, if you had read the rest of last months newsletter as thoroughly as you read Texincia, you would have seen a column devoted to the events planned for each meeting up through the end of the year.

And as for Assembly articles, well, I don't write them. I print most of what is submitted. I too would like to see more high level articles so if you know someone who is willing to contribute see that they get in touch with me.

Sincerely,
Carole Goldstein
Editor

THE MEMBERSHIP CHAIRMAN SPEAKS

At the last meeting, I was approached by a new member with his son. He wanted to know where he could receive instructions on setting up his TI computer. He had recently purchased a complete system, essentially, for use by his son. His question sort of left me wondering how our group could minister to this new member's needs. At the meeting of the Membership Committee meeting, I repeated the man's request. At that time, Sandy Bartels, a member of the Executive Board and the Membership Committee, brought forward the fact that she had recently brought in a new member who is also a good friend of her's. Sandy went on to say that her friend left her first meeting quite dismayed with the technical aspects and language of that first meeting that she attended. As she is new to the TI computer, she is in need of more elementary forms of instruction. Listening to Sandy I came to some interesting conclusions:

The first conclusion that I came to was that as the group has gotten older, the technical expertise of its individual members has increased greatly. As a result of our increased sophistication, it has become very easy for us to forget our initial experiences as TI users and as members of the club. (I can well remember when I could neither load or save anything in my console's buffer.)

I can still remember my first group meetings: I too didn't understand much of what was going on. I didn't know anybody and I was unable to get most of my questions answered. It seemed that everyone knew someone else. It also seemed that there were meetings going on within the general meeting. I was indeed very frustrated. Fortunately, I remained patient. The longer that I attended the meetings, the more people that I began to meet and know. Later, I too began to know individuals who could answer my questions or direct me to someone who could. As I can still remember my initial experiences as a new member, I can very easily appreciate the problems and needs of the new members in the group. Merely because there are so many sophisticated and knowledgable users, within the group, we must still minister to the needs of the new members who have yet to attain the degree of sophistication and knowledgability that most older members have achieved. To make a long story short, new and beginning members ARE important and must continue to remain important to our club. As we are attempting achieve greater public visibility and to involve owners of TI computers who have "closeted" their machines, we must remain sensitive to the needs of the less sophisticated newer members.

The second conclusion that I came to was the fact that many people who have recently purchased their TI computers have purchased complete systems that, at one time, many of us could only dream about being able to afford and own at some future date. Most of the older members, like me, for whom money has always an important consideration, purchased only consoles in the beginning. As our systems improved and increased in size, so did the relative degree of our facility, knowledge, and sophistication. In other words, our systems grew parallel to and with our abilities. Many new members have purchased complete systems with software and hardware that I personally would be unable to handle without a fair amount of study and practice. People in this category have systems with two particularly great potentials: There exists the potential to do fascinating things along side the potential to confuse the heck out of the owners. I can well appreciate the needs of people in this category.

The third conclusion that I came to was the fact that we have need of an on-going educational function within our group that will meet the needs of its new members who lack the expertise of the older members but who have the desire to both acquire and improve their computer user's skills and knowledge. Our Membership Committee decided that at the very next meeting, we would establish a S.I.G. (Special Interest Group) for all members who are in need of beginning computer instruction. Our intention is to start at the very BEGINNING. The name of the S.I.G. was later suggested by Carole Goldstein. This S.I.G. will be therefore called the "...and now what do I do next?" S.I.G. The first meeting of this S.I.G. will be held immediately after the general meeting for March. It will deal with things that most older members now take for granted such as how to hook up all of the components of your system, how do you use the R.F. modulator, how do you load and save programs, etc. We then want to expand the instructions into lessons in the BASIC language. Subsequent instruction will gradually work into and include intermediate level knowledge and concerns.

Here, I would like to suggest three really fine sources of information to new members: The first is the blue instruction book which comes with all TI 99/4A consoles. It is an extrememly informative and lucid book related to T. I. BASIC. The second and third sources that I would like to bring to the attention of new members are the sets of programmed instructions titled, Teach Yourself BASIC and Teach Yourself Extended BASIC. These sets of lessons are available in either cassette tape or disk format. Some of our vendors should have them or be able to get them if there is a desire for them. As a new user, I found these lessons extremely helpful. Also, relative to the need to learn the BASIC language, I will have a sheet on the sign-in table for all members to sign who would like to have instruction in BASIC. Last year, our former president, Sam Pincus, gave lessons after the regular meetings to all members who wanted instruction in T.I. BASIC. There was no charge for this instruction. The only requirement was that those who attended the class had to bring a console and t.v./monitor. These classes worked out very well. If there is a need for similar classes, we will consider setting them up after seeing how many members request them by signing the sheet.

The fourth conclusion that I came to was that my greatest acquisition of computer knowledge came to me through my use of the the modem and through interfacing with the various bulletin boards that cater to the needs of T.I. owners/users. I therefore strongly suggest that all new members upgrade their system to the point where they are able to log-onto bulletin boards as soon as possible. (This will require a console, modem, RS-232 card, and a T.E. II cartridge or some disk based terminal emulator program.) Whenever I log-onto one of the bulletin boards, I see question which require answers, which were placed within the message bases by some users with specific needs. I also see answers and responses to those questions. I must also admit that some of the questions placed on bulletin boards were my own. I am also very grateful to those users who have taken the time to respond to my questions in the past.

The last conclusion that I came to was that there are some fairly good ways for a new member to get to know other members. One way is to attend S.I.G.'s. Next, I strongly feel that the best way to meet other members is to get involved by volunteering or attending the Executive Board meetings which are open to all. If you come enough times, I will give you a money back guarantee that you will be put on some committee. Don't be bashful. All members who want to get involved are welcome to come and participate. The Executive Board meets from 11:00 until the beginning of the regular meeting in one of the smaller rooms next to the meeting hall.

If you don't see us, just stop and listen for the sounds of laughter, fighting, arguing, and the other sounds which indicate different forms of human interaction.

The new signing in proceedures worked well at the last meeting. We will therefore continue to use them at the next one though there will be a different arrangement for the placement of the tables.

I would just like to thank the three gentlemen, who have volunteered to help me, for the great job that they did at the last meeting: Ken Knapp, Barney Kugel, and Al Stump. I really do appreciate the aid.

I will once again have out on the table a list for volunteers to help distribute letters and membership guides to TI owners at, their own expense, within their own zip code areas. By the April meeting, we will start with the distribution of these letters and guides.

That's it for now. I hope that another article by me won't be necessary next month. See you all at the meeting!

SOFTWARE REVIEWS: Jack Topham

I recently read that one item on a TI99 Christmas wish list was for a DISK CATALOG program that would print out a header with the users name, the date, and the time. I remembered a program that was on the down loads about a year ago called "DISK INDEXER 10" by Mike De Frank. Since I have the CORCOMP buffer/clock card, I added the necessary lines and have given the new program to the Library, under the title of BEST/LOAD since I do think it is a very good Extended Basic Disk Indexer and Loader. Files are displayed for the selected drive and can be RUN by selecting off the menu. An ASSY LOAD-RUN option is included. You can even add comments on a file by file basis which are printed out with the index. A neat program and the price is right. Change line 902 to YOUR name. If you dont have the clock card, just delete lines 201, 235-237, 905, 907, and 908.

Each year at this time we all must face up to the IRS and get our 1040 forms completed. To make the job easier I use a BASIC program which I revise year after year as the IRS makes their annual changes to the forms. This year they left forms A and B alone and made only two small changes to the 1040 itself which made my job a snap. The program is called INCOME/TAX and as you respond to screen prompts will calculate your tax and any refund due you. It will also print out a complete report following every line on the 1040, A, and B forms. I always send a copy along with my completed forms. The program does not come with any guarantee so check the numbers before signing the return. Use to date has not revealed any errors. Yes, its in the LIBRARY

Several times in the past I has alluded to writing or editing Basic programs in TI-WRITER and have left many of you thinking "how so?". Well TI-W uses DISPLAY/VARIABLE 80 files and BASIC MERGE format is DISPLAY/VARIABLE 163 so all we need to do is be able to convert one to the other. Several programs are available that do just this. One of the first was "PROGRAM WRITER" or PW from THE SOFTIES; the people that also offer a great Assembly language tutorial called "TUTOR".
Getting a Basic program into D/V80 is very easy. Load the program into memory and then LIST it to a DISK file and you will have a D/V80 file that can be loaded and edited in TI-W! After you finish editing or writing a new program, save to DISK as usual. This D/V80 file is now processed by RUNning PW which generates a D/V163 file which can now be MERGED into memory (XB). The program can be RUN and/or saved normally as a PROGRAM file. Really very easy. PW includes a LINE Editor for those that dont have TI/WRITER.
There have been numerous programs kicking around to do the 80 to 163 conversion. File names of some of the other ones I have are: "CONVERTER" and "80MERGE". IT works and the TI-W editor is far more powerful than BASIC. Use only one statement per line to avoid problems.

Another program the serious 99'er should have is called "EPSON". This program has also been on many of the boards and I have given the LIBRARY a copy to be sure its available. This program should be on all your DISKs that have files that use the printer since it allows you to send commands to the printer for any/all of the EPSON options. These are locked in unless you turn off the printer or send overriding commands via your program. A neat utility.

For our many new members I want to review an older product but still the

only one like it. AUTO SPELL-CHECK from DRAGONSLAYER is still the only spelling checker for the TI-WRITER. SPELL-CHECK loads automatically as Option 3 "UTILITY" on the TI-W menu. Your file is checked against 20,000 words in the main dictionary and also against as many USER dictionaries as you choose to create at about 2000 words each. Words in your text that are not matched with words in the dictionaries are displayed for disposition: DISREGARD, ADD to user dictionary, CORRECT word, PREVIOUS word, NEXT word, and VIEW the word in context. The new words are added to the designated user dictionary, then the corrected text file is saved to a user specified file name. This file may need some cleaning up if corrected words are longer than the original. Just use Reformat in TI-WRITER Edit mode to close up the lines where the new word was inserted. Equal or shorter words require nothing.
AUTO SPELL-CHECK is first class. The latest version checks a full 92 sectors of text (5000 words) against 20,000 words in less than 6 minutes. User Dictionary searches are very fast since they seldom exceed 500 words. A highly recommended utility for those that do word processing. Available from all TI99 suppliers. Yes, this review was spell checked!
The Myark RAM card allows you to keep all the TI-W files and SPELL CHECK in RAM memory for instant loading. This eliminates the need to have the drive run thru 450 sectors of words during checking. The new MG GRAM KRACKER allows yet one more step to speed and ease. All the TI-W files as well as gram memory can be changed to operate TI-W with the RAM DISK set to other than drive 1. (drive 5 in my case.) TI-W is in memory all the time and only the TEXT DATA files are saved/loaded using a disk drive. AMIGA eat your heart out.!
Since I am rambling on about TI-WRITER I will add to my reference about the CORCOMP print buffer. 64k of true PIO buffer. That is approximately 8,000 words or 12 to 15 busy pages of text. The buffer has pushbuttons for CLEAR, and COPY located on the back edge of the PEB card. Not too handy but serviceable. The buffer fills quickly and returns computer control back to the user so you can be doing other things while the printer is doing its thing. At $110 it is a good value and it has a battery operated CLOCK which is set/read with simple BASIC. With all these new NON-TI add-ons, the TI99/4A has truly grown up!

HUNTER ELECTRONICS

605 S.FAIRVIEW AV.
ELMHURST,ILL.60126 312-832-6550

OPEN BY APPOINTMENT ONLY-MARCH 1,1986

ABSENT-Sorry that we will not be in attendance at the March 1 Chicago User meeting due to a committment to our Wisconsin customers. In lieu of absence we are offering the following specials:

1.FREE SHIPPING-on any order over $20 that is called in on March 1,leaving a message on our answering machine what you want(Note:up to 100 mile range only)

2.T.I.WRITER-Yes, the top of the line for word processing/$29-TODAY ONLY(limited quantity)

3.SPEAKING MATH OR READING TEACHER LIBRARIES-ONLY $12

4.MBX VOICE RECOGNITION SYSTEM-$49(GET A FREE PROGRAM)

DISKETTES-We still have plenty for only $.75 each(guaranteed SS/DD).

March Special:PAINT N PRINT WITH EXPANDED GRAPHICS-ONLY $49(save $5)

EXPANSION SYSTEMS:Again, why spend all the time and money building your own system when you can STILL BUY A COMPLETE ONE FOR SO LOW A PRICE.MYARC/with 360K DRIVE,RS232,32K MEMORY,AND REMOTE CABLE -ONLY $499(extra drive only $109 plus a cable kit)

MYARC-

CORCOMP-SAME FEATURES AS MYARC, BUT ONLY $480(NOTE-NOT COMPATIBLE WITH THE NEW COMPUTER WHEN RELEASED.

TEA READERS-call for our complete price sheet and newsletter!

PRICE REDUCTIONS:MYARC DISK CONTROLLER-$159,EXTRA MEM.CHIPS-MYARC-$7 EACH

ADVANCE ASSEMBLY ROUTINE
BY NICK IACOVELLI JR

DIRECT KEYBOARD ACCESS

INFORMATION RECIEVED AND COMPILED FROM THE EXPLORER BOOK FROM MILLERS GRAPHICS PAGE 100
ALPHA LOCK ROUTINE BY A KEYBOARD SHEET FROM TEXAS INSTRUMENT.

My next article on learning assembly will be written as soons as i find a easy way to explain memory storage and not confuse you like i did on article 3. The following article is for advance asseemblers only they wanted to know how the fctn shift x routine work in FAST-TERM (FREEWARE BY PAUL CHARLITON).If you understand assembly then read the article else go read that hot to trot lady TEXCINIA LUBBOCK column.I wonder if he,she,them or "IT" wrote a column this month.

The keyboard is map through the cru lines.TI give us a routine in in assembly call KSCAN. I found that most of TI routine are slow (exspecially gplink).The following is a better way to access characters from the keyboard and joystick port.

KEYBOARD,JOYSTICK MEMORY MAP

COLUMN	0	1	2	3	4	5	6	7
ROW 7 (01)	=	.	,	m	n	/	JOY1	JOY2 FIRE
ROW 6 (02)	SPACE	l	k	j	h	;	JOY1	JOY2 LEFT
ROW 5 (04)	ENTER	0	i	u	y	p	JOY1	JOY2 RIGHT
ROW 4 (08)		9	8	7	6	o	JOY1	JOY2 DOWN
ROW 3 (10)	FCTN	2	3	4	5	1	JOY1	JOY2 UP
ROW 2 (20)	SHIFT	s	d	f	g	a		
ROW 1 (40)	CRTL	w	e	r	t	q		
ROW 0 (80)		x	c	v	b	z		

In order to access the keyboard the column number would have to be set up on the cru line.

EX. To read column 7 joystick port 2

```
LI R1,>0700     COLUMN 7
LI R12,>0024    KEYBOARD MATRIX ADDRESS
LDCR  R1,3      SEND THE NUMBER 7 FOR COLUMN 7 TO CRU LINES
                NOW YOU ARE READY TO READ.
LI R12,>0006    ROW 7 TO 0 TO BE READ IN
STCR R1,8       READS THE KEYS.
INV R1          CRU LINE MUST BE INVERTED.
```

R1= THE ROW LINES IN HEX

NOW WE DO A COMPARE ON R1. R1 WILL HAVE VALUE FROM 0 TO 255

```
IF R1=     THEN
  00       NOTHING BEEN HIT
  01       FIRE BUTTON BEEN HIT
  02       JOYSTICK  LEFT
  03       JOYSTICK LEFT AND FIRE BUTTON BEEN HIT
  04       JOYSTICK RIGHT
  05       JOYSTICK RIGHT AND FIRE BUTTON
```

There is 255 values R1 could be and i am not going to map them all; out. A good study of the map and the above example will show all Possible combinations.

If TI return the value on the cru lines for it basic CALL JOYSTICK routine the response time on the joystick will be very fast.
(boy if i every get a gram-kracker will i write a fast x-basic.)

The fctn shift x routine in CHARLITION XMODEM reads column 0

```
*       check to see if it equal to a >03    if it is
*       then the fctn and the shift key is down.he then read port 1 for a
*       >80 if that is equal then he branches to the xmodem routine.
*
*        TI keyboard routine in 9900 assembly is very slow the following
*        assembly code is my routine to speed it up.Iry it in the gram kracker
*        tell me if it speed up.I wrote it so joysticks have priority over the
*        keys instead of the way ti wrote it.
*****************************************************************************
        DEF NSCAN
KSCANI LI R1,>0700          SETUP COLUMN TO BE SCAN 7
R       LI R12,>0024
        LDCR R1,3
        LI R12,>0006        SCAN COLUMN START
        CLR R2
        STCR R2,8           READ COLUMN
        INV R2              INV CRU LINES
        CI R2,>00FF         SEE IF KEY WAS PRESS IF NOT SCAN NEXT COLUMN
        JNE RR5
        SB @B1,R1           GET NEXT COLUMN
        CI R1,>FF00         LAST COLUMN WAS CHECK NO KEY HIT GO EXIT
        JNE R
        B @OUT2
*****************************************************************************
RR5     CI R1,>0700         JOYSTICK PORT 2 SCAN
        JNE RR1
        B @JOYST2
RR1     CI R1,>0600
        JNE RR6             JOYSTICK PORT 1 SCAN
        B @JOYST1
*****************************************************************************
RR6     LI R3,TABLE
        MOV R1,R4           COLUMN NUMBER IS SAVE IN R4
        MOV R2,R6           KEY PRESS IN MAP IS SAVE IN R6
*****************************************************************************
        CB @B1,@>8374       CHECK FOR KEY UNIT 1 + 2 IF TRUE RELOAD R3
        JEQ RR6B            WITH TABLE+4 FOR UNIT 1 +2 VALUE SKIP SCAN FOR
        CB @B2,@>8374       CRTL FCTN SHIFT KEYS
        JNE RR6A
RR6B    LI R3,TABLE+4
        CI R1,>0000
        JNE OUT6
        B @OUT2
*****************************************************************************
RR6A    CI R1,0
        JNE TT2             CHECK FOR COL 0  IF COL 0 MAKE SURE SHIFT CRTL
        ANDI R2,>0F00       OR FCTN WAS ENTER WITH ANOTHER KEY
        CI R2,0
        JEQ OUT2
*****************************************************************************
TT2     LI R12,>0024
        CLR R1
        LDCR R1,3           READ COLUNM 0 FOR SHIFT FCTN OR CRTL KEY
        CLR R12             USE TEST BIT COMMAND
        TB 7                FCTN KEY
        JEQ TT
        AI R3,2             ADD 2 TO TABLE TO GET FCTN COL
        JMP OUT6
TT      TB 8                SHIFT KEY
        JEQ TT1
        AI R3,1             ADD 1 TO TABLE TO GET SHIFT COL
        JMP OUT6
TT1     TB 9                CRTL KEY
        JEQ OUT6
        AI R3,3             ADD 3 TO TABLE TO GET CRTL COL
*****************************************************************************
OUT6    SWPB R4             MOVE COLUNM TO LEAST BIT >0700 WILL EQUAL >0007
        CI R4,0             IF R4 IS EQUAL TO COL.0 ABORT MULTPLY
        JEQ OUT4
        MPY @D40,R4         MULTIPY IT BY 32 TO GET PROPER LOCATION IN TABLE
        A R5,R3             ADD IT TO TABLE.
OUT4    LI R1,>FFFF         SET ROW NUMBER TO A -1
        ANDI R6,>FF00       GET KEY VALUE
        SWPB R6
OUT3    INC R1              R1 INC ON NO CARRY AFTER SHIFTED
        SRC R6,1            SO IF R6 =0001 R1=0 WHEN SHIFTED IT WOULD MEAN ROW 7
```

```
       JNC OUT3                              R1=1 WHEN SHIFTED IT WOULD MEAN ROW 6
       MOV R1,R7           STORE THE ROW NUMBER FOR KEY UNIT 1 + 2
       MPY @D5,R1          ADJUST TABLE VALUE
       A R2,R3
       MOVB *R3,R2
*****************************************************************************************
OUT3A  CB @B3,@>8374       CHECK FOR KEY UNIT 3. IF TRUE MAKE ALL LOWER CASE
       JEQ OUT1C           UPPER
       CB @B1,@>8374
       JEQ UNIT1
       CB @B2,@>8374       CHECK FOR KEY UNIT 1 + 2 AND ADJUST FOR PROPER
       JNE OUT1R           LOCATION
       CI R7,3             0 1 2 3 = KEY UNIT2 4 5 +6 = KEY UNIT 1
       JH  OUT2A           PROPER KEY UNIT
       JMP OUT1
UNIT1  CI R7,4
       JL OUT2A
       JMP OUT1
*****************************************************************************************
OUT1R  CLR R12
       SBZ 21
       TB 7
       JEQ OUT1            JMP IF ALPHA KEY IS UP
OUT1C  CB R2,@B97          CHECK LOWER CASE a
       JL OUT1             JMP OUT IF LOWER
       CB R2,@B122         CHECK LOWER CASE z
       JH OUT1             JMP OUT IF HIGHER
       SB @B32,R2
*****************************************************************************************
OUT1   SBO 21
       CB @>8374,@B4
       JNE OUT1A
*****************************************************************************************
       CB R2,@B13
       JEQ OUT1D
       CB R2,@B128
       JL OUT1B
       SB @B128,R2         KEY UNIT 4 MAPPING
       CB R2,@B32
       JHE OUT1BA
       JMP OUT1D
OUT1B  CB R2,@B32
       JH  OUT1D
OUT1BA AB @B128,R2
       JMP OUT1D
*****************************************************************************************
OUT1A  CB @B3,@>8374
       JNE OUT1D           KEY UNIT 3 MAPPING
       CB R2,@B128         CONTROL CODE ARE RETURN FROM 128-255
       JHE OUT2            UNIT 3 HAS NO CRTL CODE
*****************************************************************************************
OUT1D  MOVB R2,@>8375        MOVE BYTE INTO KSCAN LOCATION
OUT1DA SOCB @B32,@>837C      KEY HIT SET STATUS
       RTWP
OUT2   MOVB @BFF,@>8375
OUT2A  SZCB @B32,@>837C      NO KEY HIT SET STATUS
       RTWP
*****************************************************************************************
JOYST1 CB @>8374,@B1
       JNE NOJOY
       JMP JOYLOK
JOYST2 CB @>8374,@B2
       JNE NOJOY
JOYLOK LI R1,JTABLE           JOYSTICK TABLE
       MOVB @BFF,@>8375
       SWPB R2
       ANDI R2,>001F          MASK OUT FOR PROPER BITS
       MOV R2,R3
       SRC  R3,1             CHECK FOR FIRE
       JNC JOYU
       MOVB @B18,@>8375
JOYU   ANDI R2,>001E
       SWPB R2
JOYU1  CB *R1+,R2
       JEQ JOYHIT
       INCT R1
```

```
        CB  *R1,@B0
        JEQ NOJOY1
        JMP JOYU1
JOYHIT  MOVB *R1+,@>8377    JOYSTICK Y
        MOVB *R1,@>8376     JOYSTICK X
        JMP OUT1DA
NOJOY   MOVB @BFF,@>8375
NOJOY1  MOVB @B0,@>8376
        MOVB @B0,@>8377
        CB @B18,@>8375
        JEQ OUT1DA
        JMP OUT2A
*****************************************************************************
JTABLE      BYTE >02,-4,0       LEFT
            BYTE >04,4,0        RIGHT
            BYTE >08,0,-4       DOWN
            BYTE >0A,-4,-4      DOWNLEFT
            BYTE >0C,4,-4       DOWN RIGHT
            BYTE >10,0,4        UP
            BYTE >12,-4,4       UP LEFT
            BYTE >14,4,4        UP RIGHT
            BYTE >00,>00,>00
*****************************************************************************
*                    S                UNIT   *
*              N     H     F     L            *
*              O     I     C     T     1      *
*              N     F     T     R     +      *
*              E     T     N     L     2      *
*****************************************************************************
TABLE   BYTE  61,43,5,157,0       =
B32A    BYTE  32,32,32,32,0       SPACE
B13     BYTE  13,13,13,13,0       ENTER
D5      BYTE  >00,>05,>00,>20,0   BYTES TO FILL IN TABLE LOOKUP
B0      BYTE  >00,>01,>02,>03,0
B5      BYTE  >05,61,43,>FF,0             COLUMN 0
B97     BYTE  97,122,128,>15,0
D40     BYTE  >00,40,>00,>00,0
*****************************************************************************
        BYTE   46,62,185,155,13     .
        BYTE   108,76,194,140,12    L
        BYTE   111,79,39,143,10     O
        BYTE   57,40,15,159,9       9          COLUMN 1
        BYTE   50,64,4,178,7        2
        BYTE   115,83,8,147,2       S
B46     BYTE   119,87,126,151,4     W
        BYTE   120,88,10,152,0      X
*****************************************************************************
        BYTE   44,60,184,128,14     ,
        BYTE   107,75,193,139,3     K
        BYTE   105,73,63,137,5      I
        BYTE   56,42,6,158,8        8
        BYTE   51,35,7,179,8        3          COLUMN 2
        BYTE   100,68,9,132,3       D
        BYTE   101,69,11,133,5      E
        BYTE   99,67,96,131,14      C
*****************************************************************************
        BYTE   109,77,195,141,0     M
        BYTE   106,74,192,138,2     J
        BYTE   117,85,95,149,4      U
        BYTE   55,38,1,183,7        7
        BYTE   52,36,2,180,9        4          COLUMN 3
        BYTE   102,70,123,134,12    F
        BYTE   114,82,91,146,6      R
        BYTE   118,86,123,150,17    V
*****************************************************************************
        BYTE   110,78,196,142,15    N
        BYTE   104,72,191,136,1     H
B114    BYTE   121,89,190,153,18    Y
        BYTE   54,84,15,182,16      6          COLUMN 4
        BYTE   53,37,14,181,10      5
        BYTE   117,85,125,149,17    U
        BYTE   116,84,93,148,11     T
        BYTE   48,32,190,176,15     0
*****************************************************************************
        BYTE   47,63,186,177,16     /
        BYTE   59,58,187,154,17     :
```

```
       BYTE   112,80,34,144,11     p
       BYTE   48,41,188,176,10     0
       BYTE   49,33,3,177,19       1          COLUMN 5
       BYTE   97,65,124,129,1      a
       BYTE   113,81,197,145,18    q
       BYTE   122,90,92,154,15     z
*****************************************************************************
MYREG1 EQU >83E0
B15    EQU B97+3
B18    EQU B18A+4
B1     EQU B0+1
B2     EQU B0+2
B3     EQU B0+3
B4     EQU B4A+4
B32    EQU B32A+1
B61    EQU B5+1
B43    EQU B5+2
BFF    EQU B5+3
B122   EQU B97+1
B128   EQU B97+2
       EVEN
NSCAN  DATA MYREG1,KSCAN1
*****************************************************************************
STATUS EQU >837C
SET  DATA >2000
       DEF TEST
       REF VSBW
TEST  LWPI MYREG
TEST1 MOVB @B4,@>8374       PUT KEY UNIT BYTE IN >8374
       BLWP @NSCAN           .
       MOV  @STATUS,R3
       COC @SET,R3
       JNE TEST1
       CLR R2
       MOVB @>8375,R2
       BL @BINDEC
       JMP TEST1
*****************************************************************************
*******DEF  BINDEC        ROUTINE TO CONVERT BINARY TO ASCII DECIMAL
*******REF  VSBW                BY KEN DURIGAN
*                          MODIFY BY NICK IACOVELLI JR
ZERO   DATA 0              VALUE TO BE SHOWN IS IN R2
DTEN   DATA 10
HEX30  DATA >30
MAXCNT DATA >FFFF
SAVRTN DATA 0
MYREG  BSS  >20
R5LB   EQU  MYREG+11       REGISTER 5 LOW BYTE
BINDEC MOV  R11,@SAVRTN    RETURN ADDRESS
CLEAR  LI   R0,1           CLEAR SCREEN ROUTINE
       LI   R1,>2000       BLANK CHARACTER
CLRLUP BLWP @VSBW
       INC  R0
       CI   R0,768         HAVE I CLEARED ENTIRE SCREEN
       JNE  CLRLUP
       MOV R2,R1
       LI R0,>85           SHOW ASCII VALUE
       BLWP @VSBW
       SWPB R2
START  LI   R0,>91         MIDDLE OF FOURTH LINE IN VDP
       MOV  R2,R4
SETUP  MOV  R4,R5          SETUP DIVIDE
       CLR  R4
       DIV  @DTEN,R4       SPLIT NUMBER INTO QUOTIENT AND REMAINDER
ASCII  A    @HEX30,R5      ASCII REPRESENTATION OF REMAINDER
       MOVB @R5LB,R1
       BLWP @VSBW          WRITE ON SCREEN
NEXT   DEC  R0             GET READY TO WRITE DIGIT ON LEFT
       C    R4,@ZERO       HAS ENTIRE NUMBER BEEN DISPLAYED
       JNE  SETUP
       RT                  RETURN TO CALLER
       END
```

FROM THE OTHER ORPHANAGES

One of my tasks is to review nearly 50 newsletters from the other User Groups received each month. For two months the other groups have carried lauditory articles about our TI FAIRE, almost to the exclusion of other items of interest to us. This month things are pretty well back to order and I have some things to share with you.

From the MUSKEGON area UG comes a tip for speeding up your XBASIC programs. They suggest entering CALL LOAD(-31878,N) where N is the number of sprites used from 0 to 28. Don't forget CALL INIT before the CALL LOAD. Try it.

From the HOOSIER UG article on the FAIRE is a review of a new book telling the story of TI and our computer. "THE ORPHAN CHRONICLES" by RON ALBRIGHT, Jr. I have read the book and must say it was thoroughly delightful. A real insight to what happened, and our CUG is featured as well. Well worth the $9.95 plus $2 S/H. From Miller Graphics.

The NORTHWEST OHIO UG offered this TI TIP. To avoid a BUFFER FULL notice in the TI-WRITER, you can should save the file and then use the SD command to see the file size. Since the BUFFER is full at 92 sectors you can see right where you are. Thanx NW O UG!

From JACKSONVILLE UG in ARIZONA comes this tip from Ralph Devine. Many of us use BASIC and XBASIC in a truly unstructured way. We build the program as we go, using RES to open up lines, etc. Our program LISTs in jump No. order and finally when it is done and it RUNs OK, we are pleased. It seems that inside our 99/4A marvel our jumbled line order is preserved and time is lost looking for the next line to execute. SOLUTION! Save the file using the XB MERGE option. Sure it's slow but it is saving the program in sequential line order in MEMORY IMAGE format. Type NEW then reload using MERGE then SAVE as a PROGRAM. This can make an appreciable difference in execution time. Give it a try!

And from SEMINOLE, FLORIDA these hints:
Pressing FCTN X ten times before INITializing a DISK with TI DISK MANAGER will cause the DISK to be protected from LIST and COPY.
CALL LOAD(-31744,Y) in MINIMEMORY will change the background color. Y=16 to 31.
And, if you hit FCTN 3 instead of 2 and the line disappears, don't panic. Hit FCTN 4 then type the line number and hit FCTN X. IT'S BACK!

Oh yes. Want to LIST or COPY an XB program that was PROTECTED? Don't tell anyone but just enter CALL INIT then enter CALL LOAD(-32187,0) after the file is in memory.

The CENTRAL IOWA UG has 99 TIPS to offer for $4.50. Send to: JOHN HAMILTON, Central Iowa UG, 4228 E. Clinton Ave., Des Moines, IA 50317.

```
CALL INIT
CALL LOAD(-31744,255,255,240)
CALL LOAD(16376,97,67,72,76,78,48,0)
CALL LOAD(-31885,2,224,131,224,4,192,2,1,37,20,2,2,2,224,4,32,32,44)
CALL LOAD(-32196,0,1,76,244,0,0,0,4,32,32,36,4,91)
```

```
150 FOR C=9460 TO 9492 :: CALL LOAD(C,128):: NEXT C
```

All 'Bout C

This is the first in (hopefully) many tutorials on the C language. Bear in mind that I am *not* an experienced C programmer. In fact, c99, by Clint Pulley, is not only the first C compiler for the 4A, but is also my first exposure to the C language. This tutorial, as well as any future tutorials, will be short and sweet. No lengthy explanations. You can spend $20+ for a book on C for that. I will focus on specific problem areas that I have experience in learning C, in hopes to make the language less mysterious and more understandable to everyone interested.
As I stated above, I will not be going into detailed explanations of all facets of C. As such, I am assuming that the reader knows how to type in a C program and run the compiler to generate runable object code.

LOCAL AND GLOBAL VARIABLES

In c99, there are two basic "storage classes" of variables: local and global (more commonly called external and automatic). In a BASIC program, all variables are global: once the contents of a variable is changed, it is changed throughout. In Extended BASIC, you have the ability to write user-defined subprograms, using the SUB statement. With this feature, you can incorporate local variables; these have no effect on the outside, or calling program, even if the variable names occur elsewhere in the program. The same effect can be easily achieved in C. Any variable you use in a C program must be declared before it can be used, and is done the following way:

```
int a;          /* This defines an integer variable "a" */
char string;    /* this defines a character variable "string" */
```

Now where these declarations appear in the program determines whether they are automatic (local) or external (global). Basically, there are three basic parts to a C program: the header, the main() function, and the collection of user-created functions. The header contains, among other things, any #include statements which direct the compiler to load in other files. The header is the first thing that appears in a C program, before the main() function. If a variable declaration is included in the header, then it is an external variable. This means that you may use that variable anywere in your program, anywhere in your functions. If you change the value in one function, it changes throughout program. To create an automatic variable (local), declare it *inside* the function where it will be used. In effect, that variable will live only in its "home" and no where else in the program.
Here is a short C program that is quite useless, but illustrates the concept of automatic and external variables, as well as the three parts of a C program.

```
/* A C PROGRAM which illustrates storage classes */
/* The following is the HEADER */
#include dsk1.stdio          /* loads in a separate file i/o library */
int index;                   /* An external integer variable */
char string;                 /* An external character variable */
/* The following is the main() function */
main()                       /* start of main() function..every program must */
{                            /* contain this function! This is executed      */
/* when the program is run.                                                  */
int x;                    /* An automatic (local) integer variable */
putchar(12)               /* clears screen */
puts("Please enter your name:");
gets(string);             /* Even though "string" was not declared in the */
/* main() function, it is EXTERNAL, so it can   */
/* used anywhere. */
display();                /* Calls display(), a user-written function. */
puts("\n\nThat's all, folks!");
}
/* The following is a user-defined function */
display()
{
int boogie;               /* boogie is local to display() only. */
puts("\n\nYour name is: ");
puts(string);             /* Since string is external, it can be used */
}                         /* anywhere. */
```

As you can see, since "string" is external, I used it in 2 separate functions (main() and display()) and I only declared it once, in the header. "x", on the other hand, is declared in main(). This means I cannot refer to "x" in any other function without declaring it again. "x" lives within main() only. The same goes for "boogie"...only the display() function knows that "boogie" exists.
Some other brief points: when you *call* another function, place a semicolon after the function name. This tells the compiler that line is a statement. To *define* a function, DO NOT put a semicolon after the function name. This tells the compiler that it will be defining a new function. Furthermore, a function

body is enclosed in braces. Following this logic, you can see that main() is a function, and because there is no semicolon following the name, I am indeed defining the main() function. But main() is a very special function. When this program is compiled and run, the main() function is executed first. It is from this function that you call all other functions. It is the MAIN program. Other than this, it behaves as a normal function. The body is enclosed in brares. In fact, you can call main() just like a normal function. If, somwhere in another function, you insert a main(); (include the semicolon), the main program will run again! You can even call main() from inside main()! This is called recursion, where the main program calls itself. Look at this:

```
main()
{
puts("Recursion");
main();
}
```

This program will run forever, printing "Recursion" and then calling itself over and over again.

Well, that's it for now. I have briefly covered the difference between automatic and external variables, and how to declare each. Since the only C compiler available right now for us is c99, my comments have been limited to what can be done with that compiler. In other "true" compilers, you have many more storage classes from which to choose, and you have more flexibility in how to declare them. I also quickly covered the difference between calling a function and defining one. Admittedly, this first tutorial was VERY basic, but we all have to start somewhere! In the future, I plan to cover how to handle strings and pointers, two sticky stituations at best! If you have any comments or questions, please direct them to me, Warren Agee 70277,2063. Have a ball!

How To Function Properly

By Warren Agee 70277,2063

Second in a series of tutorials on the C language

In my first tutorial, I covered storage classes, something necessary to know before you even start programming in C. Functions are another basic concept which must be grapsed before writing C programs. Simply put, a function is a subroutine designed to perform a specified task. In some cases, values are passed to and from functions, while other functions require no communication. Numerous functions are part of the standard C library, like gets() and puts(), which allows input and output of strings, respecitively. Others, like fopen(), are kept in function libraries and stored on disk. And, of course, you may write your own functions. Indeed, the process of writing a C program involves writing user-defined functions, then putting all these functions together into a runnable program.

So, where do we begin? First of all, naming conventions. Although a function may have a name of any length, the c99 compiler only recogizes the first six characters, and they may be only alphabetic. Unlike most other compilers, the underscore (_) is not allowed. Secondly, what distinguishes a function name from a variable is the presence of parentheses. Depending on the purpose of the function, the parentheses may be empty, like getchar(). If the function requires values to be passed to it, these are placed inside the parentheses, as in puts("\nHello there!") . Now that the cosmetics are out of the way, let's get down to creating a function.

As I mentioned in the last tutorial, to call a function, merely type in its name, followed by a semicolon. To alert the compiler that you are creating a function, omit the semicolon.

```
clr()
{
int c;
putchar(12);
}
```

Here we define function called clr(). Note the missing semicolon. Also note that since the parentheses are empty, we are not going to communicate any values to the function. Next we have an opening brace, which signals the beginning of the function body. Note that the brace aligns with the first letter in the function name above; this is a standard C convention to make programs easier to read. Then we indent a few spaces, another convention. We then define the integer variable "c." Because this statement occurs inside the function body, it is local to that function (See Tutorial #1 for more info). The next statement is a standard console i/o function which prints a character to the screen whose ASCII value is in parentheses. In this case, putchar(12) simply clears the screen. We then find a closing brace which ends the function. Notice that the two braces line up.

Suppose that we want to pass one or more values to a function. Look at this:

```
add(n1,n2)
 int n1,n2;
{
int sum;
sum=n1+n2;
return(sum);
}
```

The first line tells the compiler to expect 2 values in the parenthesis when this function is called. We give these two values the names n1 and n2. When one calls this function, two numbers may appear in parentheses *like add(1,2)* or two variables *like add(a,b)*. The next line is a variable declaration, which was described in the first tutorial, but the purpose here is a little different. The function add() receives two values; now the compiler has to know what KIND (class) of values they are. Since we are passing numbers, we declare them as integers. also notice that this must come *before* the opening brace. We then declare another variable, sum, to hold the sum of the two integers. We perform the addition just as one would do in BASIC. The next line is very important.

When this function is called, we give it two numbers, and we want back the sum, right? Since the variable "sum" is local to add(), once we return to the calling program, the value of sum is lost. "Sum" only exists in add() and nowhere else. What we have to do is artificially send the value of "sum" back to the calling program, and we do this with the return statement, as shown above. Now, when we call add(), we will get back the value of sum, like this:

```
main()
{
int c;
c=add(5,2);
}
```

The expression "add(5,2)" is replaced by the answer, and we assign that value to c. If we just wrote "add(5,2)" and did not assign it to anything, the sum would just be discarded.

But why do all this? We could just declare "sum" as an external variable in main(). That way "sum" would retain it's value throughout the entire program. In very large programs, you can run into difficulties if you use only external variables. Stick to local (automatic) variables whenever possible.

Well, there you have it! There is a lot more to cover as far as functions go. The return statement only returns ONE value, no more. If you need more than one value back from the function, you have to use pointers. Pointers can be quite sticky and confusing to beginners, so I will be spending quite some time on them in the next few tutorials. So stay tuned, and experiment! It's the only way to learn! (Well, reading my tutorials may help a bit!)

How To Create A Function Library in C

Another in a continuing series of tutorials on beginner's C

By Warren Agee

(The following deals strictly with the c99 compiler, written by Clint Pulley.)

A large part of programming in C consists of using previously created functions in new programs. Such a collection is commonly called a FUNCTION LIBRARY. There are basically two ways maintain a collection of such functions: as source code, and as object code.

As Source Code

The simplest of the two methods is to keep your functions as source code. For all the examples in this tutorial, I will be referring to the CONV;C file which comes on the c99 release diskette. This is a source code file (the ";C" notation in the filename indicates this) which contains two functions: atoi and itod. Atoi converts a string into an integer (Ascii TO Integer) and itod converts an integer into a string (Integer TO Decimal string). At the present state of c99, these two functions are absolutely necessary if the programmer desires numeric input from the keyboard: the number is read in as a string, then converted to a number. To make these two functions available to a C program, the following should appear in the header (or beginning) of the program:

```
#include dsk1.conv;c
```

The #include directive tells the compiler to look for the specified file and treat it as source code to be compiled right in along with your program. This would achieve the same effect as merging the conv;c file onto the end of your program (using an editor such as TI WRITER). You may think of the #include directive to be synonymous with .IF (Include File) from TI WRITER. You may use #include in this way to load in ANY source code you may have, and as many files as you need. Although this is a simple way to keep a function library, it slows down the compiler a great deal since it has to compile the library every single time the main program compiles. A quicker way to do this is have the function library already compiled beforehand, which brings us to method 2.

As Object Code

This is by far the preferred way to maintain a function library. In essence, what you do is compile and assemble the function library beforehand so it need not be compiled every time the main program compiles. However, instead of only one file (the source code), you now need TWO files for each function library. One file is the compiled library itself, and the other is a C source code file which contains the REFs for the functions accessed in the library. Sound confusing? To make things clearer, here is a step-by-step guide on turning the CONV;C file into a compiled function library:

1. Run the CONV;C file through the compiler; when asked for "main program?", answer (N)o.

2 Assemble the resultant code, and call the new file anything you want (like CONV).I suggest not putting any suffixes (like ;C or ;0) to denote this is not a main program but a function library.
3 Go into your editor and create and save the following file:

```
#asm
REF ATOI,ITOD
#endasm
```

4 Make sure there is a space preceeding the REF.
5 Again, call this file anything you want, perhaps CONVRT.
6 To make this new function library available to any program you write, insert #include dsk1.convrt in the header of your program.
7 Once your main program is compiled and assembled, LOAD in (through E/A #3) your program, then the CONV file (the object code). The order in which you load in the files will not matter. The program can now be run as usual.
Explanation: Since the CONV file is no longer source code but object code, we cannot directly #include this file. #include only expects source code. Furthermore, since the CONV;C file is not compiled in along with the main program, the REF ATOI,ITOD tells the assembler that the functions atoi and itoa are located elsewhere, in another file (ie. external REFerences). In this case, the functions are located in the CONV object file. If you do not include the REFs, the assembler would generate errors.
What's the advantage? Initially creating the function library this way is a bit of a hassle. But in the long run, it will save you time when it comes to compiling your programs. With the first method, your function library is compiled every single time you compile your program. In the second method, the function library does not even enter the picture until you actually load and run the program.
To summarize, to create an object-code function library, 1) Compile the source code and assemble. 2) Create a separate file with REFs containing the names of all the functions in the library. 3) #include this file in your main program. 4) When Loading and Running your program, Load in the object-code function library.
Well, I hope this clears up any confusion you may have on exactly WHAT a function library is and how do I use it. To see another example, take a look at the STDIO file on the release disk. This is the file that you #include if you want to do file i/o. All it contains is REFs for the functions and some #defines for some constants. The real function library is CFIO, which is what you load in when running your program. This procedure is very different from stadard C, in which #include just loads in object code instead of source code, so I hope that this tutorial can clear up any misunderstandings anyone may have. Have fun!!

POINTERS - PART I
By Warren Agee (70277,2063)
Third in a series of articles on the C language.
Of all the aspects of the C language, pointers are the hardest for the beginner to grasp. However, once mastered, one will find that pointers are what makes C a powerful language.
Simply put, a pointer is an address, or memory location. When one declares a variable (like int c;), that variable resides somewhere in memory. A pointer to the variable "c" is the address where "c" lives. This is advantageous if we want to change a variable that is local to another function. Using pointers gives us a way to get through the barrier of being local to another function. Think of it as going through the basement to get the contents of a variable. So how do we do this?

```
int c;
int *ptr;
```

The first line just declares a normal int variable. The second line declares a "pointer" variable named "ptr." Pointer variables are preceeded with an asterisk. Now, the first line tells the compiler that we have an integer-type variable, and it's name is "c." The second line says that, first of all, we have a pointer variable. Its name is "ptr." In addition, ptr is going to point to an integer-type variable--that's what the purpose of the int in the second line. Right now, ptr does not point to anything at all. We have merely created a variable, and have told the compiler what kind of variable it will point to. Similarly, char *goose; declares a pointer variable called goose which will point to a char-type variable. Think of it this way: a pointer variable's purpose is to "look" at other variables. But you have to tell it what it is looking at...an integer or a char-type variable.
Now, if we want ptr to point to "c", we do this:

```
ptr=&c;
```

Notice that the asterisk is gone. The asterisk has two purposes, one of which is to DECLARE a pointer variable. The other purpose will come later. The "&" can be pronounced "the address of." So "&c" means the address of c. This statement assigns the address of c to ptr. If we now do c=5, what will ptr contain?? The same thing. ptr holds the location of the variable c. No matter what c contains, the location of c will not change. Variables cannot move around in memory. Ptr just contains a number, perhaps 15000, just a memory location. To tell ptr to look somewhere else, say the variable x, all you need do is ptr=&x.
Now is the time to make an important distinction:

```
int *ptr;     /* ptr is a pointer variable */
ptr=&c;       /* &c is a pointer constant  */
```

You can change the contents of a pointer variable. You cannot change a pointer constant--it is a number! Just like you can say x=3 but you cannot say 3=x. This may seem obvious, but this can get confusing later on. Just remember the difference between a pointer variable and a pointer constant. The first is a variable, the second is a number. A pointer variable contains a pointer constant, but you can use constants in other places as well. More on that some other time!!

Now that we know how to declare a pointer variable and assign it, what do we do with it??. Well, look at the following:

```
c=5;
*ptr=5;
```

The first line is obvious; it assigns c the value of 5. But what does the second line do?? The same thing!! Here we are using a technique called "indirection," or, as I like to call it, going through the basement. ptr contains the address, or location of c. If you were to print the contents of ptr, you would have some large number. But once we put the asterisk in front of it, we are saying "look at ptr's address, and access what is sitting there." In this case, we are saying, "Ptr, you are looking at a variable. Put the 5 there." You are making two jumps at once...the compiler looks at the address in ptr, then jumps to that address and see what's there. Similarly, if we want to know the value at c, we can do this:

```
int d;
d=*ptr;
```

Get the address out of ptr, hop over, get the value sitting there, and assign it to d. We are accessing the variable c INDIRECTLY, by using it's address. This seems like an awfully silly way to do things!! Why all this hanky-panky with pointers and go DIRECTLY to the variable in question? Look at this:

```
int *ptr;         /* declares an external pointer to an int */
main()
{
int answer;    /* automatic (local) integer */
ptr=&answer;   /* ptr now points to answer */
add(5,2);      /* calls add() */
}
add(n1,n2)       /* n1=5, n2=2 */
int n1,n2;       /* declares the above as integers */
{
*ptr=n1+n2;
}
```

This itsy-bitsy program combines several things I have covered before. Take a good look at the pointer used. First of all, we only have one external variable here: ptr. If we were to move ptr inside main(), that would make it unavailable to add(). So we declare it as external. Then we declare answer to be an int. Now, using the address operator (&), assign the address of answer to ptr. Now that we have done this, we can access answer anywhere in the program. Then we call the add() function. Once inside, we add the two numbers together, and, using the indirection operator (*), we tell the compiler, "Here is this sum. Go to the address contained in ptr, and deposit this sum there." When we exit this function and go back to main(), where does the sum end up? Why in answer, of course! Ptr contained the address of "answer." In fact, you can think of the compiler as a mailman. He looks inside ptr, gets the address, and delivers sum to the mailbox it found at that address...in this case, that mailbox is the variable "answer."

Note that in the above example, we used ptr to point to only one variable. Suppoed we want several answers, and we want to keep them in separate variables? All you need do is change the contents of ptr to point to whatever variable you want, like this:

```
ptr=&answer1;
(...)          /* calculate answer */
ptr=&answer2;
(...)
ptr=&answer3;
(...etc.)
```

Just by changing the contents of ptr, you can point to any variable you want. The above examples are trivial. From the last article, you learned how to easily return a value back to the calling function using the return() statement. But return() only gives back one value. By using pointers, you can alter as many values as you want. For instance, supposed you want to swap the contents of two values. This would be done like this:

```
main()
{
int x,y;
x=2;
y=19;
switch(&x,&y);
}
switch(n1,n2)
int *n1,*n2;
```

```
;
int temp;
temp=*n1;
*n1=*n2;
*n2=temp;
}
```

X and y and local variables. Using normal means, we cannot change the values of x and y outside of main(). So, instead of giving add() just the 2 variable on a platter, we give them the addresses. In this way, add() can go through the basement and change the contents of x and y. So, in order to inform switch() that it is getting addresses (or pointer constants), we declare n1 and n2 to be pointer variables. Only pointer variables can contain addresses& n1 and n2 now hold the addresses of x & y. We create a "temp"orary variable, and we do the switch. Since n1 and n2 are pointer variables, tg get at the actual values, we use indirection (*). If we had just n1=n2 instead of *n1=*n2, all we would be switching are addresses, but not the contents of the addresses. Just a pointer variable by itself holds an address. But with an asterisk, we access the value contained at that address.

The main thing to remember here is that you can pass values to functions easily. But in order to CHANGE the value of an outside variable, you must use pointers.

Wow'' Confusing, isn't it?! I suggest you reread this tutorial many times. Buy a book on C (a good one) and read all you can about pointers. I've tried to make things a bit clearer by using "ordinary" language (like "through the basement"). When fiddling with numbers and pointers, you will run into difficulty seeing your results because c99 does not have printf(), which allows the output of numbers. In our case, we must first convert the number into a string, then print out the string. This is done with the file called CONV;C on the release disk. Please refer to the file called CONVT.C in this DL for a little tutorial on how to use the CONV;C function to print out numbers. Next time, I'll cover char arrays and strings, and, eventually, the biggie, string arrays.

TECH CORNER

```
; P O U R   M A N S   D O U B L E
;
;     D E N S I T Y   D I S K
;
;       C O N T R O L L E R
;
; or how to get 360K bytes per drive
; using the TI Disk Controller Card
; and 96 tpi double side drives.
; (eg TEAC 55f, TANDON TM100-4
;
;*****************************************
;*
;* This code, when substituted for the
;* existing DSR code, converts disk #4
;* and disk #5 into 40/80 track inter-
;* lace mode and disk #9 to 40/80 track
;* non-interlace (single side). The
;* modification has been completely
;* compatable with all software tested
;* including P-System, Disk Manager,
;* Editor Assembler and Basic. Note
;* however that files can only be
;* exchanged with "normal" format disks
;* using the disk copy routines or dsk3
;* (disk #9).
;*
;* This code is placed into the public
;* domain by the author for non comm-
;* ercial use.
;*
;* Any questions may be directed to the
;* author at
;*                Andy Cooper
;*                121 Clearview Drive
;*                Downingtown Pa 19335
;*
;* or Compuserve 71016,1743
;*    Delphi     andy4820
;*
;*****************************************
;
        clr     r7                      ; existing code
        ci      r1,     1440            ; 1440 is max sector #
```

```
        jhe     $+152                   ; jump to error if greater
        ci      r1,     1               ; check for sector = 0
        jh      $5                      ; jump if not
        bl      @4524h                  ; if sector = 0 then restore
$5      clr     r0
        cb      @004Ch(r9),@4BA6h       ; compare drive # to 3
        jl      $10                     ; jump if 1 or 2
        div     @441Eh, r0              ; If drive = 3 use 9 sectors per track
        jmp     $20
$10     div     @547Ch, r0              ; if drive < 3 use 18 sectors per track
$20     swpb    r0
        inv     r0                      ; R0 msb contains inverted track #
        bl      @4614h                  ; set up vdp write
        movb    r0,     @0FFFEh(r15)    ; Store new (calc) track #
        movb    r0,     @5FFEh          ; write trk # to 1771 disk controller
        ci      r1,     9               ; check for sector > 8 (side two)
        jl      $30
        ai      r1,     -9              ; If >9 subtract nine and....
        sbo     7                       ; select side two (head 2)
        li      r7,     0100h
$30     swpb    r1
        inv     r1                      ; R1 msb contains inverted sector #
        movb    r1,     @5FFCh          ; write to 1771 sector register
        jmp     $40                     ; waste a word !!!!!!
$40     cb      r0,     @5FF2h          ; Back to original code.
```

```
;*****************************************
;*
;* To install a 2732 or 2732A prom on
;* the TI Disk controller card with the
;* above changes perform the following
;* steps.
;*
;* Remove the ROM at location U26.
;*
;* On the back (non component) of the
;* board cut the wide (+5v) etch between
;* U26 pins 21 and 24.
;*
;* On the front of the board cut the
;* wide etch going to U26 pin 21.
;*
;* On the front (component side) of the
;* board cut the etch going to U26
;* pin 18.
;*
;* NOTE: to ensure that the etch is cut
;* I recommend that two cuts 1/16 inch
;* apart be made and the copper trace
;* be removed between the two cuts.
;*
;* Install a 24 pin socket at U26
;*
;* On the back (non component) side of
;* the board add the following wires:
;*
;* U26 pin 18 to U26 pin 20.
;*
;* U27 pin 18 to U26 pin 21.
;*
;* Via hole in center of chip area
;* between U26 pin 21,22 to junction
;* of C26 and U36 pin 24 (+5v).
;*
;* Install prom in socket and.....
;*
;* thats all!
;*
;*****************************************
```

```
;*****************************************
;*
;* The following is an Editor Assembler
;* program that may be used to convert
;* normal disks to interlace format
;* prior to changing the Disk DSR. If
;* you wish drive 3 can be used to copy
;* single side disks to drives 1 and 2
;* using the P-System filer or Disk
;* Manager. (adapted from sector RW
```

```
;* tutorial writen by Tod Kaplan).
;
MYREG   BSS 32
        DEF START
        REF DSRLNK,VSBW,VMBW
        REF VMBR,KSCAN
START
        LWPI MYREG
LOOP    CLR @>8374
        BLWP @KSCAN
        CB   @KKEY,@SPACE
        JNE  LOOP
        CLR  R4
        CLR  R5
        CLR  R6
LOOP1   CI   R4,360     ;720 FOR DOUBLE SIDE
        JL   LOOP2
        LIMI 0
        LIMI 2          ;WAIT FOR QUIT
        JMP  LOOP1
LOOP2   MOV  R4,@SECTOR ;SECTOR TO PROCESS
        BL   @RSECT     ;READ SECTOR
        MOV  R5,@SECTOR ;SECTOR TO WRITE
        BL   @WSECT     ;WRITE SECTOR
        INC  R4         ;INCREMENT READ SECTOR (LINEAR)
        MOV  R4,R1
        CLR  R0
        DIV  @NINE,R0   ;CALC TRACK
        MOV  R0,R2      ;TRK TO R2
        ANDI R2,1
        JEQ  EVEN       ;JMP IF EVEN TRACK
ODD
        CLR  R2
        MOV  R0,R3      ;TRK TO R3
        DIV  @TWO,R2    ;DIVIDE IT BY 2
        CLR  R3
        MPY  @NINE,R2
        LI   R2,711
        S    R3,R2
        A    R1,R2
        MOV  R2,R5      ;INTERLACE (SIDE 2) TRK/SECTOR
        JMP  LOOP1
EVEN
        CLR  R2
        MOV  R0,R3      ;TRACK TO R3
        DIV  @TWO,R2    ;DIVIDE TRACK BY 2
        CLR  R3
        MPY  @NINE,R2
        A    R1,R3
        MOV  R3,R5      ;SIDE 1 TRACK
        JMP  LOOP1

*********************************************
* READ SECTOR ROUTINE
*
RSECT
        MOV  @DUMPAB,R0
        LI   R1,DUMDAT
        LI   R2,2
        BLWP @VMBW
        MOV  @KD1RD,@DRVFLG ;READ FROM DRIVE 1
        MOV  @DUMBUF,@BUFADD
        MOV  @DUMPAB,@PABADD
        BLWP @DSRLNK
        DATA 10
        RT
*********************************************
* WRITE SECTOR ROUTINE
*
WSECT
        MOV  @DUMPAB,R0
        LI   R1,DUMDAT
        LI   R2,2
        BLWP @VMBW
        MOV  @KD2WR,@DRVFLG  ;WRITE TO DRIVE 2
        MOV  @DUMBUF,@BUFADD
        MOV  @DUMPAB,@PABADD
        BLWP @DSRLNK
        DATA 10
        RT

**********************
* DEFINES AND DATA
*
 NINE    DATA 9
  TWO     DATA 2
  KKEY    EQU >8375
  PABADD  EQU >8356
  BUFADD  EQU >834E
  SECTOR  EQU >8350
  DRVFLG  EQU >834C
 SPACE   DATA >2020
  DUMPAB  DATA >1500
  KD1RD   DATA >0101
 KD1WR   DATA >0100
KD2WR   DATA >0200
KD3WR   DATA >0300
DUMDAT  DATA >0110
 DUMBUF  DATA >1000
         END
```

RAM BUGS?: Texincia Lubbock

Gee, wiz, it was great to be in New Orleans, (make sure you get the accent right), but it certainly is good to be back in town again. There's something about he Windy City that makes you homesick when you're away too long.

Seems like I have created a bit of a contraversy lately. Why even the bearded wonder himself, *DAVE HAKELY*, was making reference to the fact he thought I might have spent some time in a Women's correctional facility. I think you're thinking about the wrong women, *Dave*. I wouln't think of steping outside the law.

Is it true that the programming contest has been changed to a WOLFMAN *IACOVELLI* look-alike contest? Or is it a *BUZZ KRANTZ* look-alike contest?

Hey, *Buzz*, thanks for the early Valentines Day message on the Board... And I love you too! Don't think for a minute that I haven't noticed you, but please remember to send a dozen leather roses on February 14th to complete the message.

What's this I hear about *STEPHAN MEYERS*, announcing he is desirous of buying an Atari. How could you possibly think of deserting your own computer? After all, I'm sure there wouldn't be as big a software library available to you. I thought you had better taste! And besides, your talents are put to the best use on the TI.

This issue of the CHICAGO TImes celebrates the beginning of its 4th year of publication! Congrats are in order to all who have contributed to the longevity of this wonderful publication. *RYTE DATA*'S news letter is still around, too. In their December issue, they talk of expanding into a magazine format. In doing so,they are offering an advance, lifetime subscription for just $50.00! Now I ave to figure out if that's a better deal than HCM'S offer. RYTE DATA is also ooking for knowledgable technical people to go live there (Haliburton, Ontario, Cananda) and work for them... I would voice my opinion as to who I might like to see sent up there, but my fingers have been getting me into a little too much trouble lately.

Speaking of HCM, where is it? Is this chronicle going to bite the dust? A close friend of mine informs me that she no longer has a desk there and is wondering if she's going to have her job much longer. Well, I must admit I'm sorry to see them go, at least for sentimental reasons. I used to get a great deal of information and joy out of the publication when it was known as 99'er Home Computer Magazine. Unfortunately, *KAPLAN* did not maintain his support of the TI. Instead, he acted like a rat and everybody knows that rats are the first to jump from a sinking ship. Well, the 99/4A ship is still afloat and the rats are in deep water. Call it poetic justice. In any case, we must start giving greater support to those publications, who, like MICROPENDIUM, have not wavered in their support of our machine.

Where was *TODD* (you sure have a nice bod) *KAPLAN* hiding out at the last meeting? A thorough search the Fireside Lounge failed to route this wonderful hunk. Sorry I missed him, I was going to give him a copy of ORPHAN CHRONICLES, but now I've decided to keep it for myself.

See you at the next meeting. Don't forget to get your taxes done. Hope it's a refund and a big one.

CHICAGO TI USER'S GROUP
PO BOX 578341
CHICAGO, IL 60657

FIRST CLASS MAIL
DATED MATERIAL
DO NOT DELAY

REMarks: Carole Goldstein

Well, this puts another issue of the CHICAGO TImes to bed. Thanks again to all those who have contributed . To those of you who have complained about various parts of the newsletter and also about some of the meetings, please remember that all of us that do all this work for the User's Group are VOLUNTEERS. We do this because we enjoy it. We get personal satisfaction on various levels from our contributions. If you think something is not quite the way it should be, then become an active member and do your part to improve the group instead of sitting on the sidelines and complaining.

Two new languages have surfaced in the past few months. The library finally has its version of TI Pilot and C-99 has replaced FORTH as the most intriguing language. In this issue, I have reprinted a "C" tutorial which was downloaded off of Compuserve. We have the authors permission to do this and I hope you will all benefit from the amount of space I have devoted to it. Also in this issue is an in-depth assembly article submitted at the last minute by Nick I. This newsletter speaks many languages.

Also last minute came a note on the board from Sam Pincus that there is possibly a mistake in his article. Due to the shortness of the month and the need to get this issue to the printer early, I was unable to wait for his revision which he promised next Monday. We will let you know of the changes at the meeting and in the next issue. See you at the meeting March 1st.